UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/797,068	03/11/2004	Yih-Feng Hwang	ASH-03-010	4377

25537 7590 02/14/2008

VERIZON
PATENT MANAGEMENT GROUP
1515 N. COURTHOUSE ROAD
SUITE 500
ARLINGTON, VA 22201-2909

EXAMINER	
STEELMAN, MARY J	

ART UNIT	PAPER NUMBER
2191	

NOTIFICATION DATE	DELIVERY MODE
02/14/2008	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

patents@verizon.com

Advisory Action Before the Filing of an Appeal Brief	Application No. 10/797,068	Applicant(s) HWANG, YIH-FENG	
	Examiner MARY STEELMAN	Art Unit 2191	

--The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

THE REPLY FILED 22 January 2008 FAILS TO PLACE THIS APPLICATION IN CONDITION FOR ALLOWANCE.

1. ☒ The reply was filed after a final rejection, but prior to or on the same day as filing a Notice of Appeal. To avoid abandonment of this application, applicant must timely file one of the following replies: (1) an amendment, affidavit, or other evidence, which places the application in condition for allowance; (2) a Notice of Appeal (with appeal fee) in compliance with 37 CFR 41.31; or (3) a Request for Continued Examination (RCE) in compliance with 37 CFR 1.114. The reply must be filed within one of the following time periods:
   a) ☐ The period for reply expires _____ months from the mailing date of the final rejection.
   b) ☒ The period for reply expires on: (1) the mailing date of this Advisory Action, or (2) the date set forth in the final rejection, whichever is later. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of the final rejection.

   Examiner Note: If box 1 is checked, check either box (a) or (b). ONLY CHECK BOX (b) WHEN THE FIRST REPLY WAS FILED WITHIN TWO MONTHS OF THE FINAL REJECTION. See MPEP 706.07(f).

Extensions of time may be obtained under 37 CFR 1.136(a). The date on which the petition under 37 CFR 1.136(a) and the appropriate extension fee have been filed is the date for purposes of determining the period of extension and the corresponding amount of the fee. The appropriate extension fee under 37 CFR 1.17(a) is calculated from: (1) the expiration date of the shortened statutory period for reply originally set in the final Office action; or (2) as set forth in (b) above, if checked. Any reply received by the Office later than three months after the mailing date of the final rejection, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

NOTICE OF APPEAL

2. ☐ The Notice of Appeal was filed on ___. A brief in compliance with 37 CFR 41.37 must be filed within two months of the date of filing the Notice of Appeal (37 CFR 41.37(a)), or any extension thereof (37 CFR 41.37(e)), to avoid dismissal of the appeal. Since a Notice of Appeal has been filed, any reply must be filed within the time period set forth in 37 CFR 41.37(a).

AMENDMENTS

3. ☐ The proposed amendment(s) filed after a final rejection, but prior to the date of filing a brief, will not be entered because
   (a) ☐ They raise new issues that would require further consideration and/or search (see NOTE below);
   (b) ☐ They raise the issue of new matter (see NOTE below);
   (c) ☐ They are not deemed to place the application in better form for appeal by materially reducing or simplifying the issues for appeal; and/or
   (d) ☐ They present additional claims without canceling a corresponding number of finally rejected claims.

   NOTE: ___. (See 37 CFR 1.116 and 41.33(a)).
4. ☐ The amendments are not in compliance with 37 CFR 1.121. See attached Notice of Non-Compliant Amendment (PTOL-324).
5. ☐ Applicant's reply has overcome the following rejection(s): ___.
6. ☐ Newly proposed or amended claim(s) ___ would be allowable if submitted in a separate, timely filed amendment canceling the non-allowable claim(s).
7. ☐ For purposes of appeal, the proposed amendment(s): a) ☐ will not be entered, or b) ☐ will be entered and an explanation of how the new or amended claims would be rejected is provided below or appended.

   The status of the claim(s) is (or will be) as follows:
   Claim(s) allowed: ___.
   Claim(s) objected to: ___.
   Claim(s) rejected: ___.
   Claim(s) withdrawn from consideration: ___.

AFFIDAVIT OR OTHER EVIDENCE

8. ☐ The affidavit or other evidence filed after a final action, but before or on the date of filing a Notice of Appeal will not be entered because applicant failed to provide a showing of good and sufficient reasons why the affidavit or other evidence is necessary and was not earlier presented. See 37 CFR 1.116(e).
9. ☐ The affidavit or other evidence filed after the date of filing a Notice of Appeal, but prior to the date of filing a brief, will not be entered because the affidavit or other evidence failed to overcome all rejections under appeal and/or appellant fails to provide a showing a good and sufficient reasons why it is necessary and was not earlier presented. See 37 CFR 41.33(d)(1).
10. ☐ The affidavit or other evidence is entered. An explanation of the status of the claims after entry is below or attached.

REQUEST FOR RECONSIDERATION/OTHER

11. ☒ The request for reconsideration has been considered but does NOT place the application in condition for allowance because: See Continuation Sheet.
12. ☒ Note the attached Information Disclosure Statement(s). (PTO/SB/08) Paper No(s). _____
13. ☐ Other: ___.

Continuation of 11. does NOT place the application in condition for allowance because: Examiner disagrees with Applicant's characterization of the prior art (USPN 6,910,028 B2 to Chan et al. in view of "Detecting Faults in Chained-Inference Rules in Information Distribution Systems", by Yih-Feng Hwang)

Applicant argues that Chan fails to disclose the limitations of claim 1. . See Specification [0044 - 0052] which describes six categories of chained inference faults: inconsistency, circularity, subsumption, contradiction, redundancy, and incompleteness. [0055], analyzed to find faults (conflicts / identyfying faults). Applicant has defined faults to include conflicts.

As one example, Chan disclosed (col. 5: 42-45) "the role of the Conflict Transformer 15 is to analyze the input rulesets for conflicts (identify faults) and resolve conflicts among rules from one or more rulesets based on the user defined merge policy...includes syntax and semantics to express conflict resolution (modifying business rules based on the identified faults)..." (emphasis added)
Chan: Col. 3: 58 - col. 4: 20, merger of rules with different format Col. 4: 3, "merger of rulesets (e.g., business policies)... Col. 5: 29, merge policy M1 which defines the rules and conflict resolution prioritization schemes (identifying a scope of the integration) utilized by the system for enabling the merging of the rule sets (business rules that define software in the scope of the integration) Col. 6: 11-12, FIG. 2 is a diagram depicting the high level interaction between the various components underlying the conflict handling and assimilator service 19 for rule based knowledge systems and application Col. 6: 43-61, Conflict Transformer 15...analyze the input rulesets for conflicts and resolve conflicts... Col. 7: 48-52, Conflict Transformer 15...analyze the rules for conflicts, introduces new rules and predicates based on the specifications of the merge policy 25 Col. 8: 24-32, Partially ordered priorities...represent more recently acquired rules...or rules from more authoritative sources Col. 8: 43-46, two businesses may merge rule-sets R1 and R2
Col. 7: 44-47, backward / forward inferencing Col. 8: 67, chaining. (backward chaining / multi level top down approach)

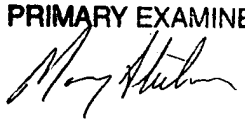
Chan disclosed (col. 5: 22) CLP...an extension of Prolog. Chan disclosed (col. 7: 44-45) backward inferencing similar to Prolog.

Inherently backward inferencing uses a depth first search. This is known in the art. Further support for this statement is attached (University of Wisconsin - Madison CS 540 Lecture Notes C. R. Dyer "Logical Reasoning Systems" Chapter 10 - see page 2).

Additionally, USPN 5,119,470 to Highland et al. (col. 1: 45-48) "Backward chaining is a depth-first search strategy of decision tree traversal. Because it is depth-first, it has the primary characteristic of focusing inferencing along reasoning paths until an answer is reached."

Examiner maintains the rejection of claims 1-28.

MARY STEELMAN
PRIMARY EXAMINER



http://pages.cs.wisc.edu/~dyer/cs540/notes/systems-logic.html

University of Wisconsin - Madison CS 540 Lecture Notes C. R. Dyer

Logical Reasoning Systems (Chapter 10)

Prolog

- Develop programming languages based on logic so that a program is a set of logic sentences and execution of a program is invoked by specifying a query (i.e., goal) to be proved
- **Prolog** is the most common logic programming language
- Prolog restricts sentences to **Horn clauses**, i.e., a sentence of the form

```
P1 ^ ... ^ Pn => Q
```

where each `Pi` and `Q` is an un-negated literal. The `Pi`'s are called the **antecedents**, or left-hand side (LHS) of the sentence, and the `Q` is called the **consequent**, or right-hand side (RHS). Syntactically written as:

```
Q :- P1, P2, ..., Pn.
```

- Horn clause types
  - **Fact**: A clause with no antecedents. In other words, just `Q`. For example, `married(Kurt, Sue)`
  - **Rule**: A clause with at least one antecedent and a consequent. For example, the following three rules together define "brother in law" in Prolog, where (universally-quantified) variables are those things that start with a capital letter.
    - brother-in-law(B,P) :- married(P, Spouse), brother-of(B, Spouse).
    - brother-in-law(B,P) :- sister-of(Sister, P), husband-of(B, Sister).
    - brother-in-law(B,P) :- married(P, Spouse), sister-of(Sister, Spouse), husband-of(B, Sister).
  - **Query**: A clause with no consequent. For example,

```
?- likes(john, X), likes(rita, X).
```

means "Is there anything that John and Rita both like?" where the variable symbol x is implicitly existentially quantified.

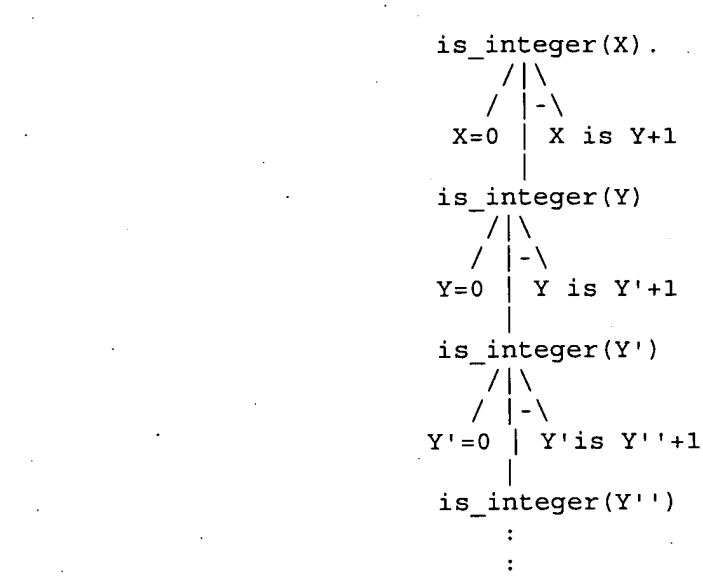
- The axioms (facts and rules) are stored in a database (the KB) that is ordered by the programmer.
- Prolog reasoner is a **backward-chaining** procedure using **depth-first search** on the ordered facts and rules until a solution is found. (Hence, it is *not* a proof by contradiction as used by resolution.)
- Example
  - DB =

```
1. b.
2. x.
3. y.
4. g :- a, b.
5. a :- x, y.
```

  - Query: `?- g.`
  - Execution proceeds as follows:
    - Unify query `g` with each of the facts, and when all fail, then with each of the "heads" (i.e., consequents) of the rules in the order in which they occur in the DB.
    - When unification succeeds (here at (4)), recursively attempt to solve the sub-goals ("sub-queries") associated with the antecedent of the selected rule. Here, these are subgoals `a` and `b`.
    - Attempting to solve sub-goal `a` uses unification again to find the first fact or rule that matches. Here, that's rule (5). This in turn creates two new (sub-sub-)goals, `x` and `y`.
    - DFS continues by attempting to solve goal `x` which in this case unifies with fact (2). By unifying with a fact, this sub-goal is "solved," so it is popped from the stack of goals to be achieved.
    - The next goal to be achieved is goal `y`. It also unifies with a fact (3), so it's solved.
    - All of the antecedents of rule (5) have now been solved, so this means that `a`, the consequent part of that rule, is solved too.
    - Finally, attempt to solve goal `b`, which succeeds because it unifies with fact (1).
    - So, all antecedents of rule (4) are solved, so it's consequent `g` is solved, and we're done. So, the answer is "Yes" `g` is entailed by the DB.
- Backtracking within the DFS for a solution is invoked in two ways:
  - Within a rule
    If antecedent `Pi` fails to be proved true, then try to prove the previous antecedent, `Pi-1`, in the current rule in a different way (i.e., using different facts or rules).
  - Use a different rule
    If a rule fails because an antecedent cannot be solved (no matter how we re-try to solve earlier antecedents), then try a different rule. That is, try to unify the current goal with the head (i.e., consequent) of a different rule.
- There is no assignment statement in Prolog because unification is used to "bind" values to variables. During backtracking variables are just re-bound.
- Example

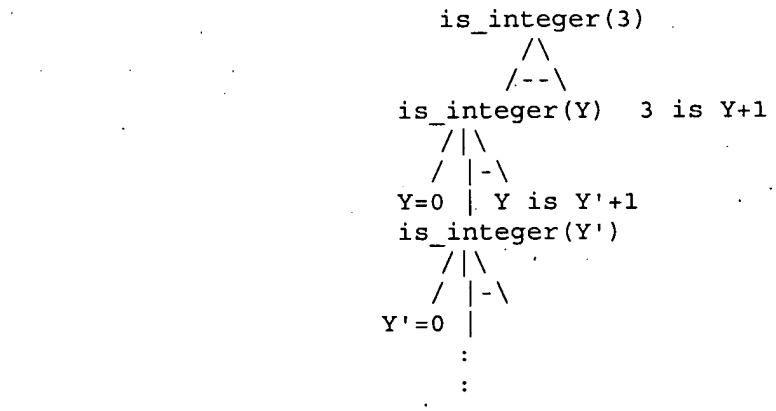
- Program for defining the non-negative integers:
  - `is_integer(0).`
  - `is_integer(X) :- is_integer(Y), X is Y+1.`
- Query for generating the non-negative integers:

```
?- is_integer(X).
```



- Query for testing if a constant is a non-negative integer:

```
?- is_integer(3).
```



Production Systems

- Invented in 1943 by Post
- Used as the basis for many rule-based expert systems
- Production System consists of 3 components:
  - **Rules**
    An unordered set of user-defined "if-then" rules of the form: `if P1 ^ ... ^ Pm then Action_1, ..., Action_n` where the `P`is are facts that determine the conditions when this rule is applicable. Each Action adds or deletes a fact from the Working Memory.

- **Working Memory** (WM)
  A set of "facts" consisting of positive literals defining what's known to be true about the world
- **Inference Engine**
  Procedure for inferring changes (additions and deletions) to Working Memory.

  ```
  while changes are made to Working Memory do:
  ```

  1. Construct **Conflict Set**
     The Conflict Set is the set of all possible `(rule, list-of-facts)` pairs such that `rule` is one of the rules and `list-of-facts` is a subset of facts in WM that unify with the antecedent part (i.e., Left-hand side) of the given rule.
  2. Apply **Conflict Resolution Strategy**
     Select one pair from the Conflict Set.
  3. **Act Phase**
     Execute the actions associated with the consequent part of the selected rule, after making the substitutions used during unification of the antecedent part with the `list-of-facts`.

- Conflict Resolution Strategies The following are some of the commonly used conflict resolution strategies. These are often combined as well to define hybrid strategies.
  - **Refraction**
    A rule can only be used once with the same set of facts in WM. Whenever WM is modified, all rules can again be used. This strategy prevents a single rule and list of facts from be used over and over again, resulting in "infinite firing" of the same thing.
  - **Recency**
    Use rules that match the facts that were added most recently to WM. Hence, each fact in WM has a time-stamp indicating when that fact was added. Provides a kind of "focus of attention" strategy.
  - **Specificity**
    Use the most specific rule, i.e., if one rule's LHS is a superset of the facts in the LHS of a second rule, then use the first one because it is more specific. In general, select that rule that has the largest number of preconditions.
- Example
  - Let WM = {A, D}
  - Let Rules =
    1. if A then Add(B)
    2. if A then Add(C), Delete(A)
    3. if A ^ E then Add(D)
    4. if D then Add(E)
    5. if A ^ D then Add(F)
  - Conflict Set = {(Rule1, (A)), (Rule2, (A)), (Rule4, (D)), (Rule5, (A,D))}

- Using Specificity Conflict Resolution Strategy, select (Rule5, (A,D)) because it matches two facts from WM while the others match only one fact each.
- "Fire" Rule5 by adding F to WM, so that now WM ={A, D, F}

- XCON (aka R1)
  - DEC's expert systems for constructing a single acceptable configuration of hardware components for a complete computer system based on partial customer specifications
  - Bottom-up, data-driven, forward-chaining deductive system for synthesizing a solution.
  - Rules are used to (1) determine if an order is complete and, if not, adds necessary items, and (2) determines the spatial relations (connectivity) of components
  - No backtracking needed --- constraints in rules are sufficient to directly construct a solution
  - Rules always determine locally whether taking a particular action is globally consistent with acceptable overall performance on the task
  - About 10,000 rules such as

```
Assign_Power_Supply_1
  IF  Most current active context is assign-power-supply
      An SBI module is in cabinet
      Position of module in cabinet is known
      Space is available in cabinet for a power supply
          for that position
      No power supply is currently available
      Voltage + frequency of components is known

  THEN  Find a power supply of that voltage and
frequency
            and add it to order
```

  - WM contains current configuration of components. For example, space filled and unfilled in cabinets and backplane slots.
  - Uses specialization conflict resolution strategy plus "context" markers in WM to control the firing of rules.
  - Groups of rules are associated into separate "contexts" which define a particular sub-task that they are used for. First antecedent in each rule indicates the context where the rule is to be used. For example, the rule above is to be used in the context of "assigning a power supply."
  - To change contexts, there are rules such as:

```
Check_Voltage_And_Frequency_1
  IF  Most current active context is
         checking-voltage-and-frequency
      There is a component requiring one voltage or
         frequency
      There is another component requiring a different
         voltage or frequency

  THEN  Enter context of
           fixing-voltage-or-frequency-mismatches

Put_UB_Module_6
```

```
        IF  Most current active context is
              putting-unibus-modules-in-backplanes-in-box
            It is known which module to insert
            The module is a multiplexer interface
                    :
                    :
        THEN  Enter context of
                verifying-panel-space-for-a-multiplexer
```

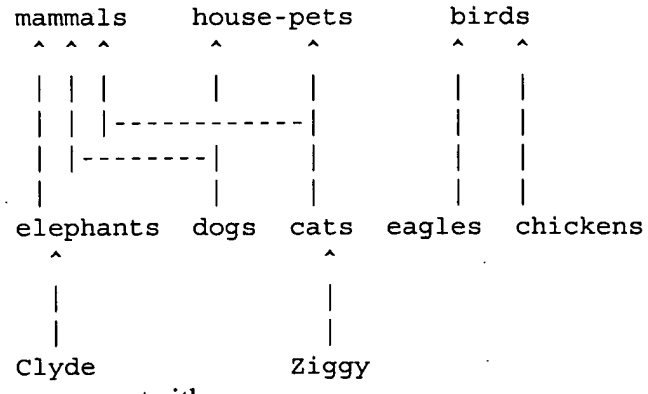
- Because of the use of the specificity conflict resolution strategy, the following type of rule is used to deactivate a context:

```
    Deactivate_Unibus_Module_Context
        IF  Most current active context is
              putting-unibus-modules-in-backplanes-in-box
        THEN  Exit context
```

- MYCIN
  System for assisting physicians in diagnosing and treating diseases caused by certain kinds of bacterial infection

Semantic Networks

- Logic and Production Systems are very fine-grained representations of knowledge, and do not allow for user to construct explicitly more complex objects. Hence, hard to define all of the primitive sentences or rules, and hard for the user to interpret the results. Also, deductive reasoning is slow because of the fine grain inferences which must be performed.
- Semantic Nets add more semantics and define specialized inference rules based on the semantics of the representation.
- A semantic network is a network of **nodes** representing basic concepts, objects, ideas, or situations, and **links** representing associations or relations between pairs of objects.
- Usually used to represent static, taxonomic, concept dictionaries
- Example



- Nodes represent either
  1. Individual objects or instances. E.g., Clyde.
  2. Classes or subclasses. E.g., mammals.
- Basic link types:
  1. **Member**
     X is a member, or instance of, of class Y. For example, "Ziggy is a cat"

could be represented in FOL by `cat(Ziggy)`. In semantic nets this would be represented by the structure:

```
                Member
    Ziggy -------------> Cats
```

4. **Subclass**
   Class X is a subclass (or subset) of the class Y. For example, "all cats are mammals" could be represented in FOL by `Ax cat(x) => mammal(x)` and in semantic nets could be represented by

```
              Subclass
    Cats --------------> Mammals
```

7. **Relation between two objects**
   Instance node X has property value Y. For example, "Ziggy is 10 years old" could be represented in FOL by `age(Ziggy, 10)` and in a semantic net by

```
                 Age
    Ziggy -------------> 10
```

10. **Relation between every instance of class X and object Y**
    For example, "all birds have 2 legs" could be represented in FOL by `(Ax) bird(x) => num_legs(x,2)`. In a semantic net this could be represented as

```
              Num_Legs
    Birds -------------> 2
```

13. **Relation between every element of class X and some element of class Y**
    For example, "everyone likes some foods" could be represented in FOL by `(Ax) person(x) => (Ey) food(y) ^ likes(x,y)`. In a semantic net this would be

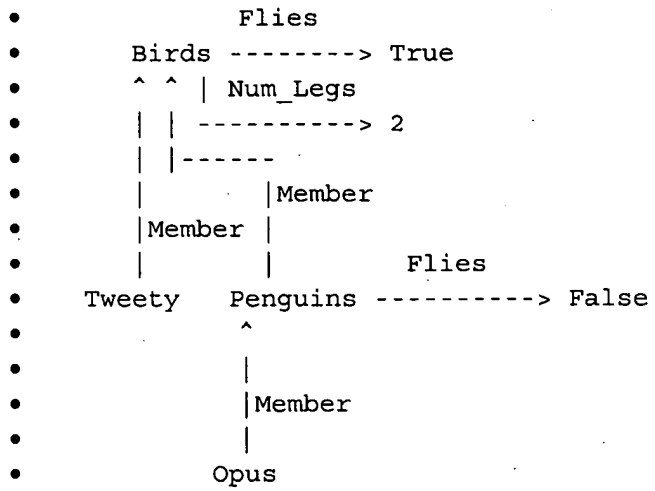
```
                 Likes
    People ------------> Foods
```

- **Inference by Inheritance**
  Objects are organized hierarchically in classes and instances of classes. Exploit transitivity of subclass (subset) relation. Inheritance is a mechanism for sharing behavior or properties that are common to a collection of classes or individuals. "Virtual Copy"

  To answer the query "How many legs does Tweety have?" we first find the `Tweety` node in the network and then check if there is a `Num_Legs` link directly from that node. If there is, then we have an immediate answer to the query. If there is not such a link, then follow the `Member` link to the class `Birds`. Now check if there is a `Num_Legs` link from that node. If there is, then we look up the answer there; if not, then follow a `Subclass` link to a superclass such as `Animals`, and continue this process until either an answer is found or else there are no more `Subclass` links to chain through in the network.

- **Inheritance with Exceptions**
  In many applications it is not possible to define relations on classes that are *necessary* properties for all instances of that class. For example, by definition all quadrilaterals have four sides, so this property is necessary for every member of that class. However, much of the time we can only say that a property is *typical* or

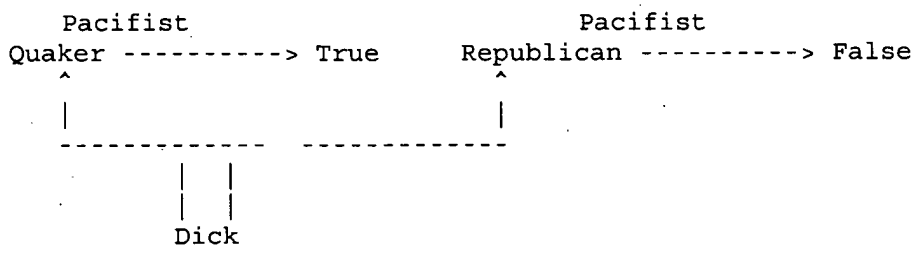
a *default* property for the instances in a class. For example, the property that birds fly is usually the case, but is certainly not required for all instances of that class since penguins and ostriches don't fly. In a semantic network, this means that we must be able to "cancel" properties of a superclass. This is done as shown in the following example:



In this case inheritance searches for the *closest* node where the property is found and uses that information to answer the query. For example, asking "Does Opus fly?" finds the property `Flies` at the `Penguins` node and therefore the `Flies` property at the superclass `Birds` is implicitly cancelled. On the other hand, asking "How many legs does Opus have?" finds the answer "2" at the `Birds` node.

- **Multiple Inheritance**
  In general, a node can have any number of superclasses that contain it, enabling a node to inherit properties from multiple "parent" nodes and their ancestors in the network. If this is the case, how should inference by inheritance work? The following rules are generally used to determine inheritance in such "tangled" networks where multiple inheritance is allowed:
  - If nodes A and B are both ancestors (through member and subclass links) of node X, and A is an ancestor of B, then B is **inferentially closer** to X than A. So, if A and B have a value for property P, X will inherit its value for property P from B (and completely ignore the value for P at A). For example, in the network above, `Penguins` is inferentially closer to `Opus` than `Birds`, so Opus inherits the `Flies` property from `Penguins`, not `Birds`.
  - If A is not an ancestor of B, and B is not an ancestor of A, then A and B are in conflict as far as X's inheritance is concerned. So, if A and B have different values for property P, then X will not inherit a value for P from either one.

For example, the query "Is Dick a pacifist?" cannot be answered in the network below.



- Nodes that are not ancestors of X are entirely irrelevant as far as X's inheritance is concerned.
